Programación Python para Machine Learning

Lección 3: Tratamiento de valores perdidos.

eip ESCUELA INTERNACIONAL POSGRADOS

ÍNDICE

Lección 3: Tratamiento de valores perdidos.

1. Valores perdidos

Desafortunadamente, los datos procedentes de problemas en el mundo real no son perfectos. En la mayoría de los casos suelen contener valores que dificultan enormemente el trabajar con ellos. Se pueden diferenciar dos tipos de estos: valores faltantes y valores atípicos. En el primer caso, la razón por la que los conjuntos de datos pueden tener valores pueden ir desde las observaciones que no fueron registradas hasta la corrupción de los datos. La importancia de su manejo reside en que muchos algoritmos de Machine Learning no admiten el trabajar datos con valores perdidos.

Objetivos

- Entender la problemática de la presencia de valores perdidos.
- Identificar de modo adecuado los valores perdidos.
- Aprender a utilizar las técnicas de eliminación de valores perdidos.
- Manejar las técnicas principales para la imputación de valores perdidos.

1.1. Marcar valores perdidos

El primer paso cuando se sospecha de la presencia de valores perdidos es su detección en el conjunto de datos. Es posible utilizar los gráficos y/o estadísticas de resumen para ayudar a identificar los valores faltantes o que están corruptos.

Una vez cargado el conjunto de datos en un DataFrame, se examinan las estadísticas de resumen de cada atributo. Los valores perdidos se indican con frecuencia mediante entradas fuera de rango; quizás un número negativo (por ejemplo, -1) en un campo numérico que normalmente sólo es positivo, o un 0 en un campo numérico que normalmente nunca puede ser 0. Se dan casos específicos, donde posiblemente haya habido un preprocesamiento previo, donde los datos faltantes se han codificado con un carácter especial, o directamente, con un espacio en blanco.

En todos los casos, se hace necesario marcar dichos valores de manera inequívoca en el DataFrame para que su procesamiento sea el correcto. En su filosofía, Pandas trata *None* y *NaN* como elementos indicativos de la presencia de valores perdidos o nulos. Para facilitar esta convención, hay varias funciones útiles para detectar, eliminar y reemplazar valores nulos en un Pandas DataFrame.

```
from pandas import read_csv
import pandas
import numpy as np

filename = 'mammographic_masses.data'

col_names = ['BI-RADS','Age','Shape','Margin','Density','Severity']
data = read_csv(filename, names=col_names)

data[data == '?'] = np.nan
```

Para verificar la existencia de valores perdidos en un Pandas DataFrame, se hace uso de las funciones `isnull()` y `notnull()`. Ambas funciones ayudan a comprobar si un valor es *NaN* o no. Para identificar valores nulos en un DataFrame, la función `isnull()` devuelve un Series de valores booleanos que son True para aquellos valores NaN.

Este Series puede usarse a modo máscara para examinar las instancias con valores perdidos, incluso para hacer un conteo de los mismos.

```
from pandas import read_csv
import numpy as np

filename = 'mammographic_masses.data'

col_names = ['BI-RADS','Age','Shape','Margin','Density','Severity']
data = read_csv(filename, names=col_names)

data[data == '?'] = np.nan

print(data.isnull().sum())
mask=data['Margin'].isnull()
print(data[mask])
```

En el caso de `notnull()` el uso sería justo el contrario. La función `notnull()` devuelve un Series de valores booleanos que son False para aquellos valores NaN y True en el caso contrario. Este Series puede usarse a modo máscara para examinar las instancias que no tienen valores perdidos. Su utilidad reside en que, a partir del resultado, se podría operar con las variables numéricas.

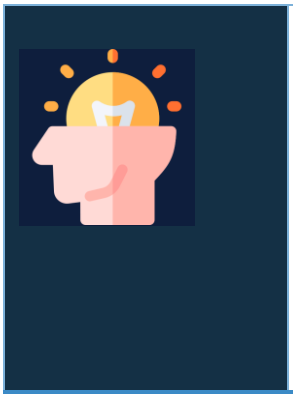
```
from pandas import read_csv

import numpy as np

filename = 'mammographic_masses.data'

col_names = ['BI-RADS','Age','Shape','Margin','Density','Severity']
data = read_csv(filename, names=col_names)

data[data == '?'] = np.nan
print(data.isnull().sum())
mask=data['Age'].isnull()
print(data[mask])
mask=data['Age'].notnull()
data.Age[data['Age'].notnull()].astype(int).describe()
print(data[mask])
```



No te olvides…

"Los datos perdidos no son raros en los conjuntos de datos de problemas del mundo real. De hecho, la probabilidad de que falte al menos un valor aumenta a medida que aumenta el tamaño del conjunto de datos."

Kuhn, M., & Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. CRC Press.

2. ESTRATEGIAS PARA TRATAR CON LOS VALORES PERDIDOS

La presencia de valores perdidos en un conjunto de datos es habitual en los datos procedentes de problemas del mundo real. Por desgracia, la mayoría de las técnicas de modelado predictivo no pueden manejar los valores perdidos. Por lo tanto, una vez detectada la existencia de instancias con valores perdidos, se debe abordar su tratamiento antes del modelado. Hay dos estrategias generales de tratamiento. Una estrategia básica para utilizar conjuntos de datos incompletos es descartar filas y/o columnas enteras que contengan valores perdidos. Sin embargo, aún incompletos esta opción conlleva el precio de perder datos que pueden ser valiosos. De todos modos, es una alternativa a valorar. Una segunda estrategia podría ser imputar los valores faltantes, es decir, inferirlos a partir de la parte conocida de los datos.

2.1. Eliminar instancias o características con datos perdidos

Eliminar las dimensiones del problema que contienen un valor faltante se trata de la estrategia más sencilla para tratar la presencia este tipo de datos. Es posible llevar a cabo este procedimiento creando un nuevo Pandas DataFrame con las filas que contienen valores perdidos eliminados. Pandas proporciona la función `dropna()` que permite eliminar tanto las columnas como las filas con datos perdidos. Concretamente, `dropna()` presenta las siguientes alternativas de uso:

- Eliminar filas con al menos un valor faltante.

```
from pandas import read_csv
import numpy as np

filename = 'mammographic_masses.data'

col_names = ['BI-
RADS','Age','Shape','Margin','Density','Severity']
data = read_csv(filename, names=col_names)

data[data == '?'] = np.nan
print(data.shape)
clean_data = data.dropna()
print(clean_data.shape)
```

- Eliminar las filas en las que todos los valores son faltantes.

```
dropna(how='all')
```

- Eliminar las columnas con al menos un valor faltante.

```
dropna(axis = 1)
```

- Eliminar las columnas con todos los valores faltantes.

```
dropna(how='all', axis=1)
```

- Eliminar filas/columnas cuando al menos haya 'n' valores faltantes.

```
dropna(thresh=2)
dropna(axis = 1, thresh=2)
```

Pasar como parámetro `inplace=True` cambia el propio DataFrame de origen. Puede resultar útil con determinados tamaños de DataFrame donde duplicarlo trae consigo problemas de memoria.

2.2. Imputación de valores perdidos

En primera instancia, se debe considerar que es posible imputar los datos ausentes. El termino imputación se refiere al uso de una técnica para reemplazar los valores perdidos. Es viable utilizar de diversos modos la información del conjunto de entrenamiento para, en esencia, estimar del mejor modo posible los valores ausentes. Existen muchas opciones a considerar a la hora de sustituir un valor perdido: un valor constante que tenga significado dentro del dominio, como el 0, distinto de todos los demás valores; un valor de otro registro seleccionado; un estadístico representativo de la columna (media, mediana o moda); o incluso un valor estimado por otro modelo predictivo.

Cualquier imputación realizada en el conjunto de datos de entrenamiento tendrá que ser realizada exactamente del mismo modo en nuevos futuros cuando se necesiten predicciones del modelo en cuestión. Esto debe tenerse en cuenta a la hora de elegir cómo imputar los valores perdidos.

Por ejemplo, si elige imputar con la media del atributo, este valor tendrá que ser almacenados para su uso posterior en nuevos datos que puedan presentar valores perdidos.

Un tipo de método de imputación de valores ausentes es la **imputación univariante**, que imputa los valores en la i-ésima dimensión de la característica utilizando sólo los valores no perdidos en esa dimensión de la característica. En este sentido, por un lado, para rellenar los valores nulos en un conjunto de datos, Pandas ofrece las funciones `fillna()`, `replace()` e `interpolate()`, que sustituyen los valores NaN por algún valor propio. La función `interpolate()` se utiliza básicamente para rellenar los valores NA en el DataFrame haciendo uso de varias técnicas de interpolación clásicas.

Sustitución por un valor constante: `fillna()`, `replace()`

```
from pandas import read_csv
import numpy as np
from sklearn.impute import SimpleImputer

filename = 'mammographic_masses.data'

col_names = ['BI-RADS','Age','Shape','Margin','Density','Severity']
data = read_csv(filename, names=col_names)

data[data == '?'] = np.nan

new_data=data.fillna(0)
data.replace(to_replace = np.nan, value = 0)
```

Sustitución por el valor anterior o posterior: `fillna()`

```
df.fillna(method ='pad')
df.fillna(method ='bfill')
```

Sustitución por un valor interpolado: `interpolate()`

```
data.interpolate(method ='linear', limit_direction ='forward')
```

Por otro lado, scitkit-learn incluye la clase `SimpleImputer` que proporciona estrategias básicas para la imputación de valores perdidos. Permite igualmente la sustitución por un valor constante proporcionado, o utilizando los estadísticos (media, mediana o moda) de cada columna en la que se encuentran los valores perdidos.

```
from pandas import read_csv
import numpy as np
from sklearn.impute import SimpleImputer

filename = 'mammographic_masses.data'

col_names = ['BI-RADS','Age','Shape','Margin','Density','Severity']
data = read_csv(filename, names=col_names)

data[data == '?'] = np.nan

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(data)
newdata=imputer.transform(data)
```

La clase `SimpleImputer` también admite trabajar con datos categóricos representados como valores de cadena o categóricos de Pandas cuando se utiliza `strategy="most_frequent"`.

```
from pandas import read_csv
import numpy as np
from sklearn.impute import SimpleImputer

filename = 'mammographic_masses.data'

col_names = ['BI-RADS','Age','Shape','Margin','Density','Severity']
data = read_csv(filename, names=col_names)

data[data == '?'] = np.nan

imputer = SimpleImputer(strategy="most_frequent")
imputer.fit_transform(data)
newdata=imputer.transform(data)
```

Un enfoque más sofisticado sería la **imputación multivariante**, modelando cada atributo con valores perdidos como una función de las otras características, y utilizar esa estimación para la imputación. Lo hace de forma iterada: en cada paso, una columna de características se designa como salida (y) y las otras columnas de características se tratan como entradas (X). Se entrena un regresor en (X, y) para con el (y) conocido.

Este procedimiento se realiza para cada característica de forma iterativa, y luego se repite un número de rondas determinado. Se devuelven los resultados de la última ronda de imputación. La clase `IterativeImputer` de scitkit-learn implementa esta estrategia.

Una técnica bastante extendida para la imputación multivariante es utilizar un modelo de kNN (k-nearest neighbors). El principio es buscar los patrones que más se parezcan al que tiene el valor perdido e imputar un dato similar al de ellos. La librería scikit-learn proporciona la clase `KNNImputer` que implementa la imputación kNN.

```
from pandas import read_csv
import numpy as np
from sklearn.impute import KNNImputer
import pandas

filename = 'mammographic_masses.data'

col_names = ['BI-RADS','Age','Shape','Margin','Density','Severity']
data = read_csv(filename, names=col_names)

data[data == '?'] = np.nan

X = data[data.columns[:-1]]

imputer = KNNImputer()
imputer.fit(X)
newdata=imputer.transform(X)
```

3. Puntos Clave

En esta lección hemos aprendido:

- Entender las razones por las que la presencia de valores perdidos son un problema a tratar en los datos.
- Utilizar las herramientas que dispone Pandas y scikit-learn para identificar la presencia de valores perdidos.
- Descubrir los principios de las técnicas de eliminación de valores perdidos y cómo implementarlas con Pandas.
- Conocer los principios de las técnicas principales para la imputación de valores perdidos y y cómo implementarlas con Pandas y scikit-learn.